Claims

[c1] What is claimed is: 1. A method of a software architecture that provides high versatility and performance, the method comprising the steps of: Having two dimensions, an application dimension and a core dimension;Relating the application dimension to the different applications; Relating the core dimension to the applications dimension;Having the COREs within the core dimension related;Having the COREs within the core dimension share information and integrate the system architecture; Having the applications based on abstractions that are composed of drivers, abstraction layers and a unique CORE; and Having a plurality of terminal devices.

2. The method of claim 1 in which said CORE consists of a complex compound of software components that dynamically accepts software extensions and exchanges information with other COREs.

3. The method of claim 1 in which said CORE consists of a complex compound of software components that dynamically accepts software extensions and exchanges information with other COREs, said abstaction layers consists of a software layer the hides implementaion detail and data structures of a specific software and said module extensions are comprised of an abstraction layer and a driver.

4. The method of claim 1 in which said core dimension consists of a plurlarity of COREs connected by a bi-directional communication means.

5. The method of claim 1 in which said CORE consists of three parts;i)a CORE Engine kernel, which manages the task processing and scheduling, manages the information exchange between the CORE dimension, and manages the dynamic extensions, ii)an InterExtension Communication and logic manager which manages the CORE"s communcation and tasks with a plurlarity of extensions, iii)an InterCORE Communication manager which manages the CORE"s communication with a plurlarity of other COREs.

6. The method of claim 1 in which said abstraction layer consists of two parts;i)a CORE-Abstraction interface, which interfaces an extension with a CORE; and ii)Extension Knowledge Layer, which contains logic and knowledge about the operations of extensions.

7. The method of claim 1 in which includes the step of having an extension driver layer that consists of two parts;i)an Abstraction-Driver interface, which interfaces the Abstraction layer with a terminal device; and ii)Driver logic used to control the terminal device.

8. The method of claim 1 used for a Control and Automation Application.

9. The method of claim 1 used for an Assets Control application.

10. A method to replace a terminal device with a new terminal device using the method in claim 1 consisting of the adding the step of changing the driver for the extension.

11. A method to add a new terminal device to a system using the method in claim 1 consisting of adding the steps of: a)constructing a new extension for the terminal device; b)interfacing the new extension into the CORE; c)asking the CORE for the required data and information to handle the new extension.

12. A computer program product for software architecture that provides high versatility and performance, the computer program product comprising a computer usable medium having computer readable program code thereon, including: Program code for having two dimensions, an application dimension, and a core dimension;Program code for relating the application dimension to the different applications; Program code for relating the core dimension to the applications dimension;Program code for having the COREs within the core dimension related;Program code for having the COREs within the core dimension share information and integrate the system architecture; Program code for having the applications based on abstractions that are composed of drivers, abstraction layers and a unique CORE; and Program code for having a plurality of terminal devices.

13. The computer program product of claim 12 wherein said CORE consists of a complex compound of software components that dynamically accepts software extensions and exchanges information with other COREs.

14. The computer program product of claim 12 wherein said CORE consists of a complex compound of software components that dynamically accepts software extensions and exchanges information with other COREs, abstraction layers consists of a software layer the hides implementation

detail and data structures of a specific software and said module extensions are comprised of an abstraction layer and a driver.

15. The computer program product of claim 12 said core dimension consists of a plurlarity of COREs connected by a bi-directional communication means.

16. The computer program product of claim 12 wherein the CORE consists of three parts and has interfaces and the program code for;i)a CORE Engine kernel, which manages the task processing and scheduling, manages the information exchange between the CORE dimension, and manages the dynamic extension, ii)an InterExtension Communication and logic manager which manages the CORE"s communcation and tasks with a plurlarity of extensions, iii)an InterCORE Communication manager which manages the CORE"s communication with a plurlarity of other COREs.

17. The computer program product of claim 12 wherein the said abstraction layer consists of two parts and has interfaces and the program code for;i)a CORE-Abstraction interface, which interfaces an extension with a CORE; and ii)Extension Knowledge Layer, which contains logic and knowledge about the operations of extensions.

18. The computer program product of claim 12 wherein it has an extension driver layer that consists of two parts and has interfaces and the program code for;i)an Abstraction-Driver interface, which interfaces the Abstraction layer with a terminal device; and ii)Driver logic used to control the terminal device.

19. The computer program of claim 12 with interfaces and the program code for used for a Control and Automation Application.

20. The computer program of claim 12 with interfaces and the program code for used for used for an Assets Control application.

21. A computer program to replace a terminal device with a new terminal device using the computer program in claim 12 with interfaces and the program code for used for changing the driver for the extension.

22. A computer program to add a new terminal device to a system using the method in claim 12 with interfaces and the program code for used for: a) constructing a new extension for the terminal device;b)interfacing the new

extension into the CORE;c)asking the CORE for the required data and information to handle the new extension.